Rasenmähroboter

Aufgaben

Roboter, die selbstständig den Rasen mähen können, sogenannte Rasenmähroboter, erfreuen sich in Deutschland immer größerer Beliebtheit. Sie werden nicht ferngesteuert, sondern mähen autonom die vorgegebene Fläche in geraden Bahnen. Die meisten Systeme nutzen zum Abgrenzen der zu mähenden Fläche ein Begrenzungskabel, durch das elektrische Signale geleitet werden, die der Rasenmähroboter beim Überfahren sensorisch erfassen kann (Material 1). Erkennt er die Begrenzung, dreht sich der Rasenmähroboter in einem zufälligen Winkel vom Kabel weg und fährt erneut in einer geraden Bahn weiter, bis er wieder an eine Begrenzung stößt. Nach diesem sogenannten Chaosprinzip erreicht und mäht der Rasenmähroboter, über eine gewisse Zeit betrachtet, alle Bereiche der Rasenfläche. Der Rasenmähroboter ist akkubetrieben und lädt sich selbstständig an einer in der Rasenfläche befindlichen Ladestation wieder auf. Die abgeschnittenen Grashalme werden nicht aufgefangen, sondern fallen einfach zu Boden.



geändert nach:
https://www.einhell.de/fileadmin/_processed_/2/d/csm_blog-guide-robot_lawnmowers_in_spring-content-2_a5fa8f13f0.jpg (abgerufen am 15.09.2020).

1 Der Hersteller eines Rasenmähroboters plant ein Simulationsprogramm zu entwerfen, um den Fahralgorithmus des Rasenmähroboters zu testen und zu verbessern. Es haben zwei Personengruppen Zugriff auf dieses Simulationsprogramm: Die Entwickler oder Entwicklerinnen (im Folgenden Entwickler genannt) und die Simulatoren oder die Simulatorinnen (im Folgenden Simulatoren genannt). Die Entwickler erstellen neue Algorithmen oder optimieren bereits vorhandene Algorithmen. Die Simulatoren können Simulationen entwerfen, durchführen und löschen. Um eine Simulation durchzuführen muss sie zunächst entworfen werden, sofern sie noch nicht vorhanden ist. Zum Entwerfen einer Simulation ist es zwingend nötig, vorher einen Rasenmähroboter und eine Rasenfläche anzulegen. Außerdem können die Simulatoren vor dem Durchführen einer Simulation einen Algorithmus importieren.
Die zentrale Klasse `Simulation` ist vorgegeben (Material 2). Eine Simulation kann mehrere Algorithmen verwalten, während der einzelne Algorithmus in mehreren Simulationen verwendet werden kann. Die abstrakte Klasse `Algorithmus` beinhaltet die Informationen über den Namen des Algorithmus und den Namen des Erstellers sowie einen Wahrheitswert, ob sich der Algorithmus derzeit im Einsatz befindet. Des Weiteren ist ein statisches Attribut `idAlgo` vom Datentyp int enthalten, welches als Objektzähler fungiert. Zusätzlich liegt ein Attribut vom Typ String mit dem Bezeichner `algorithmus` vor, in dem die Fahrbefehle des Algorithmus verkettet als String gespeichert werden. Die Klasse `Algorithmus` enthält Bewegungsmethoden, also je eine void-Methode für die Vorwärts- und Rückwärtsbewegung und je eine Methode für das Links- bzw. Rechtsdrehen. Die Methoden für die Vorwärts- und Rückwärtsbewegung erhalten als Parameter einen int-Parameter `zeitS` für die Angabe der Fahrzeit in Sekunden. Die Drehmethoden bekommen beim Aufruf einen int-Parameter mit einem Winkel übergeben. Zuletzt zählt noch die Methode `pruefeHindernisVorne` zu den Fahrbefehlen. Diese überprüft, ob sich vor dem Rasenmähroboter ein Hindernis befindet und gibt einen Wahrheitswert zurück. Eine weitere Methode `zufallswinkel` gibt nach Aufruf einen int-Wert zwischen 0 und 180 Grad zurück. Die zwei Klassen `rekursiverAlgorithmus` und `iterativerAlgorithmus` erben von der Klasse `Algorithmus`. Für rekursive Algorithmen wird in einem Attribut die Rekursionstiefe gespeichert. Für iterative Algorithmen wird in einem Attribut die Anzahl der Schritte gespeichert.

1.1 Entwickeln Sie unter Berücksichtigung der obigen Beschreibung ein aussagekräftiges Use-Case-Diagramm (Anwendungsfalldiagramm). Bestimmen Sie dabei die menschlichen Akteure, die Beziehungen zwischen Akteuren und Anwendungsfällen und eventuell vorhandene include- bzw. extend-Beziehungen.

(10 BE)

1.2 Ergänzen Sie das UML-Klassendiagramm in Material 2 entsprechend der in Aufgabe 1 stehenden Beschreibung. Modellieren Sie dazu geeignete Attribute, Datentypen, Multiplizitäten, Methoden und Navigierbarkeiten im Design-Modell.

(10 BE)

1.3 Im gegenwärtigen Zustand wurde im System noch kein Algorithmus erstellt. Der Rasenmähroboter mit dem Namen „EasyCutter42“, der Typennummer „1337“, mit vollem Akku, wobei die Integer-Werte 0-100 als Prozentwerte zu betrachten sind, und einer Messeranzahl von 3 wurde jedoch bereits angelegt. Zusätzlich existieren folgende zwei Rasenflächen:

– Rasenfläche1: Länge 10 m, Breite 55 m, eingefasst
– Rasenfläche2: Länge 4 m, Breite 9 m, offen

Die Simulation trägt den Namen „EpsilonP4“ und wurde von Edwin B. Budding am 05.02.2021 erstellt.
Überführen Sie den oben beschriebenen Sachverhalt in ein Objektdiagramm.

(7 BE)

1.4 Implementieren Sie die Klassen `Simulation`, `Rasenmähroboter` und `Rasenfläche` gemäß dem UML-Klassendiagramm in Material 2 so, dass zum Anlegen von Rasenmährobotern und Rasenflächen die Methoden `rasenmähroboterAnlegen(…)` und `rasenflächeAnlegen(…)` verwendet werden und mindestens je eine Instanz der Klassen `Rasenmähroboter` und `Rasenfläche` erzeugt wird.

Hinweise: In der Klasse `Simulation` sind die Attribute, der Konstruktor und die von Ihnen benötigten Methoden zu implementieren. Die beiden Instanzen sind selbst zu erstellen. Get- und set-Methoden sind generell nicht zu erstellen.

(13 BE)

2 Auf der Hauptplatine des Rasenmähroboters ist es an einer Stelle nötig, seriell eintreffende Binärdaten in parallele Binärdaten umzuwandeln. Zur Konvertierung der Daten (seriell zu parallel) wird als Grundbaustein das in Material 3 dargestellte Bauteil „MC74HC112A“ benötigt.

2.1 Analysieren Sie das Bauteil und beschreiben Sie dessen Funktion.

(6 BE)

2.2 Dokumentieren Sie die Umwandlung des in Material 3 gezeigten Flipflops hin zu einem D-Flipflop schriftlich und als Schaltung.

(5 BE)

2.3 Entwerfen Sie in Material 4 unter Verwendung der dort zu sehenden Bauteile ein 4-Bit-Schieberegister mit seriellem Eingang und parallelem Ausgang.

Hinweis: Die Pins für V_{CC} und GND müssen nicht belegt werden.

(7 BE)

2.4 Der Datenstrom kann auch weiterhin seriell abgegriffen werden. Geben Sie den Zeitpunkt und den Ausgang an, an dem der serielle Abgriff am Schieberegister möglich ist.

(2 BE)

2.5 Die Wahrheitstabelle in Material 5 zeigt die benötigte Logik zum Auslesen der parallelen Binärdaten. Vereinfachen Sie die Ausgänge q_{2n+1}, q_{1n+1} und q_{0n+1} durch die Verwendung von KV-Diagrammen.

(6 BE)

3 Ein Assembler auf einem Mikrocontroller im Rasenmähroboter steuert das Mähwerk, an dem an einer drehenden Scheibe drei kleine Klingen angebracht sind, die die Grashalme schneiden. Für die Rotationsbewegung wird ein Gleichstrommotor verwendet. Solange der Mikrocontroller auf Pin 0 ein HIGH-Signal erhält, ist das Mähwerk eingeschaltet. Im normalen Mähbetrieb wird am Motor der Rechtslauf aktiviert und der Rasenmähroboter fährt vorwärts. Zum Zwecke der Reinigung des Rasenmähroboters von Schnittgut im Mähwerkbereich steht ein Reinigungsmodus zur Verfügung. Um den Reinigungsmodus zu aktivieren, erhält der Assembler auf Pin 6 über einen internen Zähler, der die Betriebsstunden zählt, ein kurzes HIGH-Signal. Ist dies der Fall und das Mähwerk ist noch eingeschaltet, wird der Rechtslauf zunächst ausgeschaltet und der Mikrocontroller wartet 10 Sekunden, bis das Mähwerk zum Stillstand gekommen ist. Danach wird am Motor der Linkslauf aktiviert, um die Verschmutzungen von den Messerklingen zu entfernen. Endet der Reinigungsmodus nach 15 Sekunden, wird der Linkslauf abgeschaltet und der Mikrocontroller wartet 10 Sekunden. Sollte Pin 0 kein HIGH-Signal erhalten, werden sowohl Links- als auch Rechtslauf deaktiviert und das Mähwerk ist ausgeschaltet.

Port 1 des Mikrocontrollers							
Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2	Pin 1	Pin 0
	↑				↓	↓	↑
	Reinigungs-modus				Rechts-lauf	Linkslauf	Mähwerk ein

3.1 Erläutern Sie die Portbelegung für Ihren im Unterricht verwendeten Mikrocontroller und zeichnen Sie ein entsprechendes Technologieschema.

(4 BE)

3.2 Stellen Sie die beschriebene Mikrocontroller-Steuerung für die Mähfunktion inklusive Reinigungsmodus des Rasenmähroboters in einem Struktogramm dar.

Hinweis: Sämtliche erforderliche Verzögerungsschleifen können als gegeben angenommen und als Unterprogramme (z. B. warten20 ms etc.) verwendet werden.

(10 BE)

3.3 Implementieren Sie Ihr Struktogramm aus Aufgabenteil 3.2 bzw. die beschriebene Mikrocontroller-Steuerung aus Aufgabe 3 in der von Ihnen im Unterricht verwendeten Assemblersprache.

Hinweis: Die als Unterprogramme zur Verfügung gestellten Verzögerungsschleifen müssen nicht implementiert werden.

(10 BE)

3.4 Um den Rasenmähroboter zukünftig noch „smarter" zu machen, soll eine intelligente Ladungssteuerung verbaut werden. Aus diesem Grund ist es notwendig zu wissen, ob sich der Rasenmähroboter aktuell auf der Ladestation befindet (Material 1). Dazu soll die Ladestation mit einem zusätzlichen Sensor ausgestattet werden.

3.4.1 Erläutern Sie das generelle Funktionsprinzip eines kapazitiven Sensors.

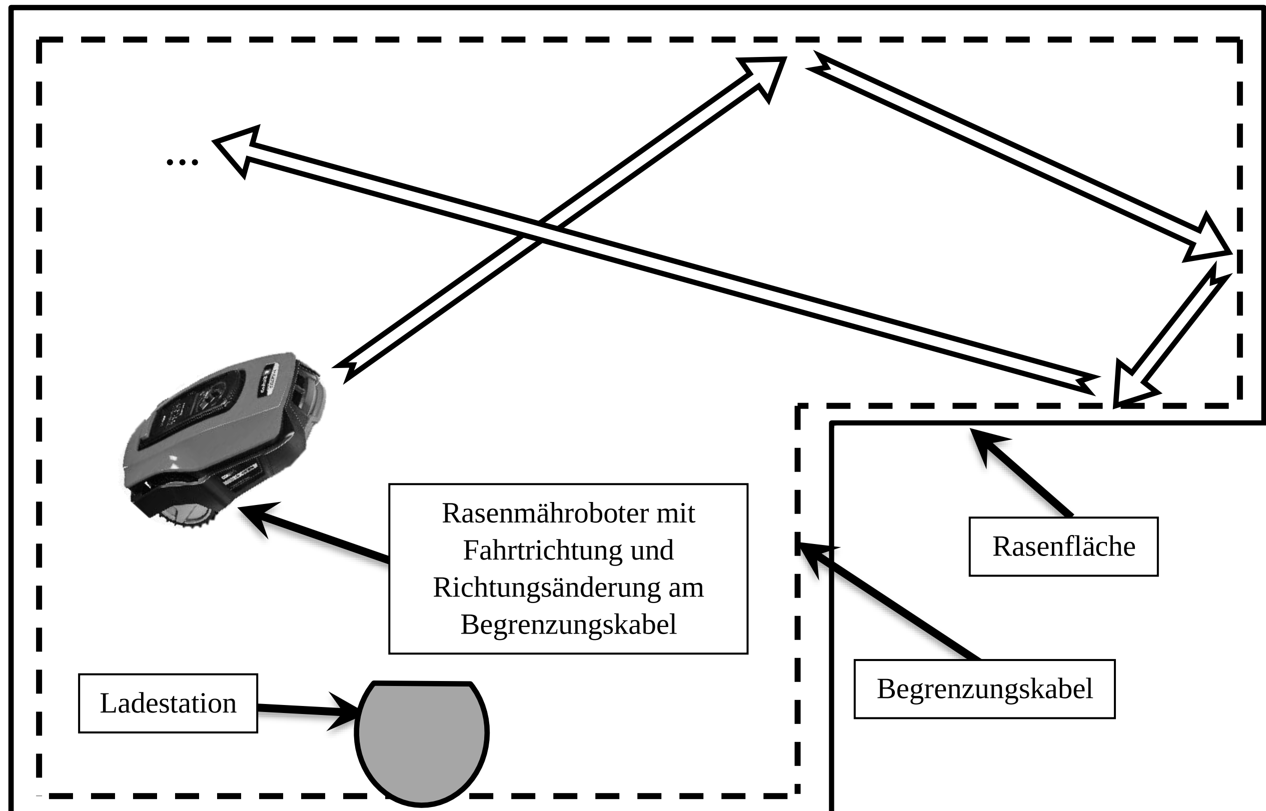
(4 BE)

3.4.2 Entwickeln Sie mit einer Ihnen aus dem Unterricht bekannten Sensortechnik ein Konzept für die genannte Anforderung an den Sensor und skizzieren Sie dieses Konzept.

(6 BE)

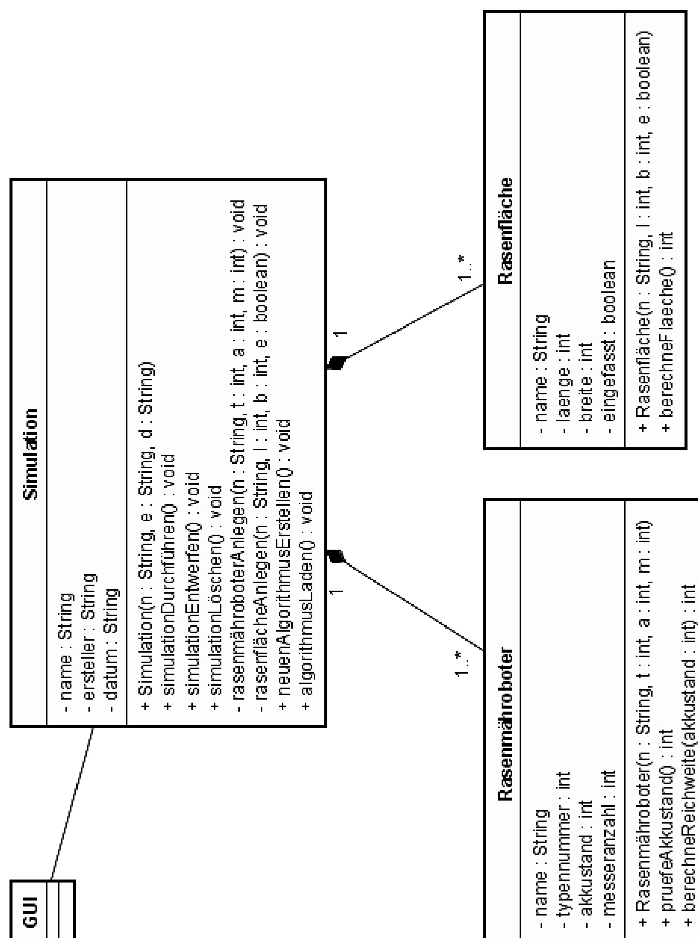
Material 1

Schaubild Funktion Rasenmähroboter



Material 2

UML-Klassendiagramm



Material 3

Datenblatt Flip-Flop MC74HC112A

MC74HC112A

Dual J-K Flip-Flop with Set and Reset

High–Performance Silicon–Gate CMOS

The MC74HC112A is identical in pinout to the LS112. The device inputs are compatible with standard CMOS outputs; with pullup resistors, they are compatible with LSTTL outputs.

Each flip–flop is negative–edge clocked and has active–low asynchronous Set and Reset inputs.

The HC112A is identical in function to the HC76, but has a different pinout.

Features

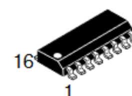
- Output Drive Capability: 10 LSTTL Loads
- Outputs Directly Interface to CMOS, NMOS, and TTL
- Operating Voltage Range: 2.0 to 6.0 V
- Low Input Current: 1.0 μA
- High Noise Immunity Characteristic of CMOS Devices
- In Compliance with the Requirements Defined by JEDEC Standard No. 7A
- Similar in Function to the LS112 Except When Set and Reset are Low Simultaneously
- Chip Complexity: 100 FETs or 25 Equivalent Gates
- These are Pb–Free Devices



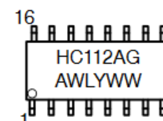
ON Semiconductor®

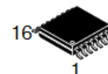
http://onsemi.com

MARKING DIAGRAMS

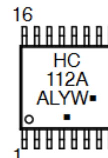


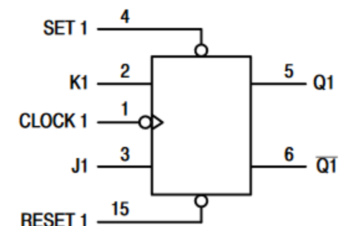
SOIC–16
D SUFFIX
CASE 751B





TSSOP–16
DT SUFFIX
CASE 948F





FUNCTION TABLE

Inputs					Outputs	
Set	Reset	Clock	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	L*	L*
H	H	↓	L	L	No Change	
H	H	↓	L	H	L	H
H	H	↓	H	L	H	L
H	H	↓	H	H	Toggle	
H	H	L	X	X	No Change	
H	H	H	X	X	No Change	
H	H	↑	X	X	No Change	

*Both outputs will remain low as long as Set and Reset are low, but the output states are unpredictable if Set and Reset go high simultaneously.

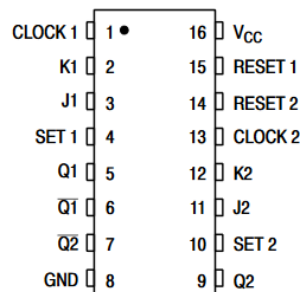


Figure 1. Pin Assignment

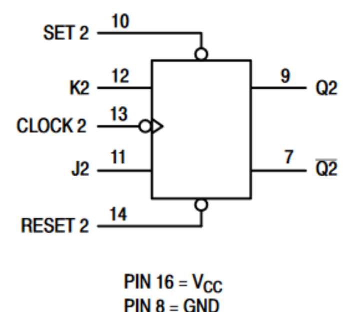
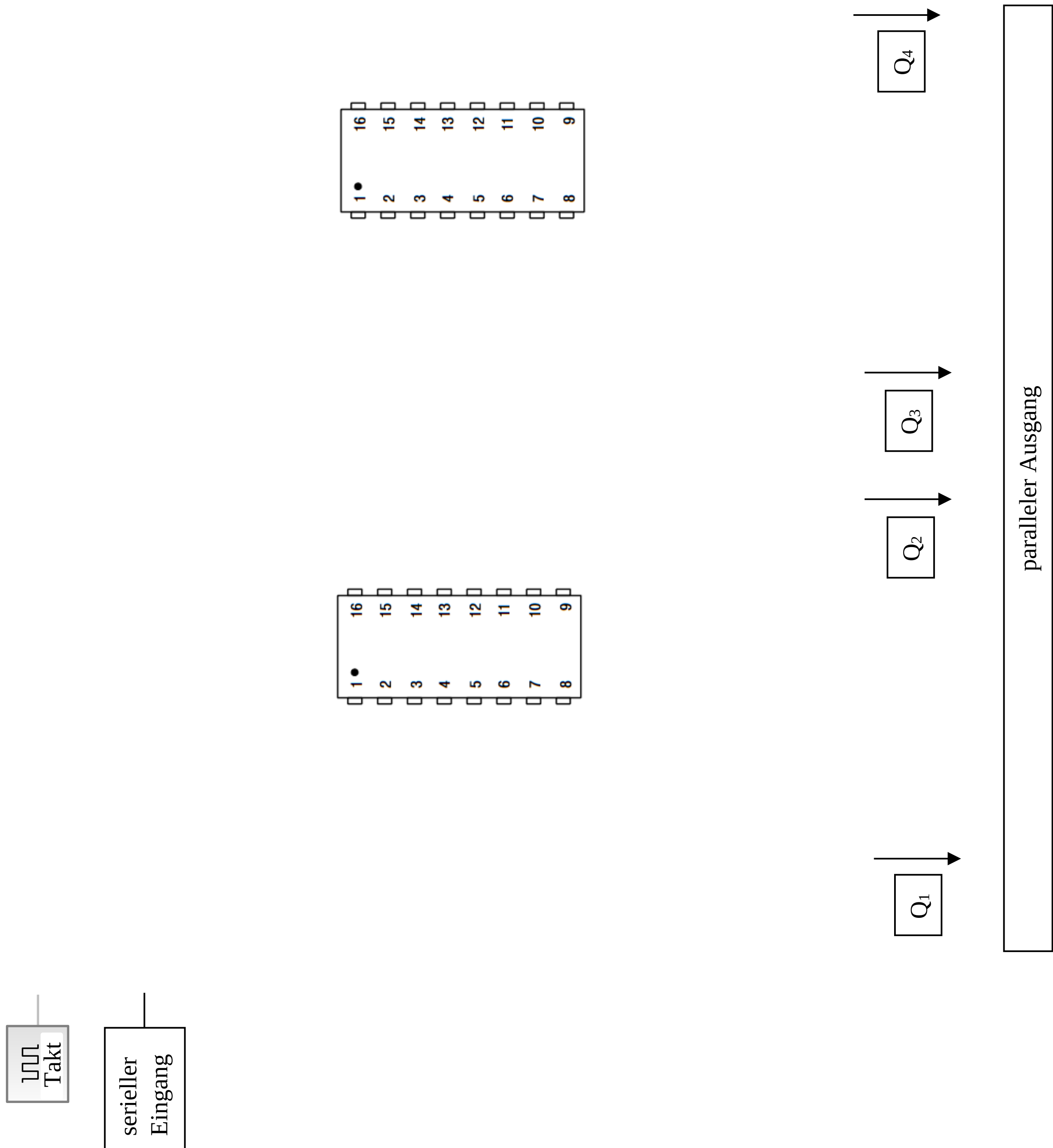


Figure 2. Logic Diagram

https://www.onsemi.com/pub/Collateral/MC74HC112-D.PDF (abgerufen am 12.01.2021).

Material 4

Schaltung Schieberegister



Material 5

Wahrheitstabelle für paralleles Auslesen

Index	q_{2n}	q_{1n}	q_{0n}	q_{2n+1}	q_{1n+1}	q_{0n+1}
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	0	0	0
5	1	0	1	X	X	X
6	1	1	0	X	X	X
7	1	1	1	X	X	X